

**Appendix A:**

This appendix contains the method developed in [5] for computing the 9-point DCT given by

$$A(n) = \sum_{k=0}^{9-1} a(k) \cos \left[ \frac{\pi}{2 \times 9} (2k+1)n \right] \quad \text{for } 0 \leq n \leq 8$$

Method for the 9-point DCT:

$$\begin{aligned}
c_1 &= -0.866025, c_2 = 0.939693, c_3 = -0.173648, c_4 = -0.766044 \\
c_5 &= 0.5, c_6 = -0.342020, c_7 = -0.984808, c_8 = -0.642788 \\
d_1 &= a(3) + a(5), d_2 = a(3) - a(5), d_3 = a(6) + a(2), d_4 = a(6) - a(2) \\
d_5 &= a(1) + a(7), d_6 = a(1) - a(7), d_7 = a(8) + a(0), d_8 = a(8) - a(0) \\
d_9 &= a(4) + d_5, d_{10} = d_1 + d_3, d_{11} = d_{10} + d_7, d_{12} = d_3 - d_7, d_{13} = d_1 - d_7, d_{14} = d_1 - d_3 \\
d_{15} &= d_2 - d_4, d_{16} = d_{15} + d_8, d_{17} = d_4 + d_8, d_{18} = d_2 - d_8, d_{19} = d_2 + d_4 \\
m_1 &= c_1 d_6, m_2 = c_5 d_5, m_3 = c_5 d_{11}, m_4 = c_2 d_{12}, m_5 = c_3 d_{13} \\
m_6 &= c_4 d_{14}, m_7 = c_1 d_{16}, m_8 = c_6 d_{17}, m_9 = c_7 d_{18}, m_{10} = c_8 d_{19} \\
d_{20} &= a(4) - m_2, d_{21} = d_{20} + m_4, d_{22} = d_{20} - m_4, d_{23} = d_{20} + m_5 \\
d_{24} &= m_1 + m_8, d_{25} = m_1 - m_8, d_{26} = m_1 + m_9 \\
A_0 &= d_9 + d_{11}, A_1 = m_{10} - d_{26}, A_2 = m_6 - d_{21}, A_3 = m_7, A_4 = d_{22} - m_5 \\
A_5 &= d_{25} - m_9, A_6 = m_3 - d_9, A_7 = d_{24} + m_{10}, A_8 = d_{23} + m_6
\end{aligned}$$

Thus, the 9-point DCT requires only 10 multiplications and 34 additions, respectively.

## Appendix B: Source Code

```

//***** ****
// File name: mdct.cpp
//***** ****

#include <iostream.h>
#include <iomanip.h>
#include <math.h>
#include <time.h>

#pragma hdrstop
#include <condefs.h>
#include "dct_lib.h"

USEUNIT("dct_lib.cpp");
-----
int borg[100036];
double
bin[100036],*ptbin,bout0[36],bout1[36];
-----

#pragma argsused
int main(int argc, char* argv[])
{
int k,n;
int i,cnt;
double diff;
clock_t ck1,ck2;

cout<<"welcome to mdct test\n";
build_costab();

// the random data
randomize();
for(k=0;k<100036;k++)
borg[k]=random(255);
for(k=0;k<100036;k++)
bin[k]=(double)
borg[k];

// verify the MDCT and the fast methods
cnt=100000;
ptbin=bin;
cout<<"Verifying MDCT3618 and MDCT3618F,
cnt=<<cnt<<endl<<endl;
for(i=0;i<cnt;i++)
{
    MDCT3618 (ptbin,bout0);
    MDCT3618F(ptbin,bout1);
    ptbin++;
    for(n=0;n<18;n++)
    {
        diff=((bout0[n]-
bout1[n])<.0000000001)?0:99999;
        if(diff>0)
        cout<<diff<<setw(10)<<bout0[n]<<setw(10)<
<bout1[n]<<"\n";
    }
}
// verify the IMDCT and the fast methods
ptbin=bin;
cout<<"Verifying      IMDCT1836      and
IMDCT1836F, cnt=<<cnt<<endl<<endl;
for(i=0;i<cnt;i++)
{
    IMDCT1836 (ptbin,bout0);
    IMDCT1836F(ptbin,bout1);
    ptbin++;
    for(n=0;n<36;n++)
    {
        diff=((bout0[n]-
bout1[n])<.0000000001)?0:99999;
        if(diff>0)
        cout<<diff<<setw(10)<<bout0[n]<<setw(10)<<bo
ut1[n]<<"\n";
    }
}

cnt=100000;
// computation time of MDCT3618
cout<<"Computation time of MDCT3618     : ";
ptbin=bin;
clk1=clock();
for(i=0;i<cnt;i++)
{
    MDCT3618 (ptbin,bout0);
    ptbin++;
}
clk2=clock();
cout<<setw(6)<<(clk2-clk1)<<endl;

// Computation time of MDCT3618F
cout<<"Computation time of MDCT3618F   : ";
ptbin=bin;
clk1=clock();
for(i=0;i<cnt;i++)
{
    MDCT3618F(ptbin,bout1);
    ptbin++;
}
clk2=clock();
cout<<setw(6)<<(clk2-clk1)<<endl;

//Computation time of IMDCT1836
cout<<"Computation time of IMDCT1836   : ";
ptbin=bin;
clk1=clock();
for(i=0;i<cnt;i++)
{
    IMDCT1836 (ptbin,bout0);
    ptbin++;
}
clk2=clock();
cout<<setw(6)<<(clk2-clk1)<<endl;

// Computation time of IMDCT1836F
cout<<"Computation time of IMDCT1836F : ";
ptbin=bin;
clk1=clock();
for(i=0;i<cnt;i++)

```

```

        {
            IMDCT1836F(ptbin,bout1);
            ptbin++;
        }
    ck2=clock();
    cout<<setw(6)<<(ck2-ck1)<<endl<<endl;

    return 0;
}
//*****dct_lib.cpp*****
//*****dct_lib.h*****



#include <iostream.h>
#include <math.h>
#pragma hdrstop

#include "dct_lib.h"

-----
#pragma package(smart_init)

// some pre-computed consine tables
double Cos99[9][9];
double Cos99W[9]; // used for DCT99W
double Cos1818[18][18];
double Cos1818F[18];// DCT1818F
double Cos3618[18][36]; // shared for
mdct3618, imdct1836
double Cos3618F[18];
double Cos1836F[18];

=====
// the COS tables for various size
=====
void build_costab(void)
{
    int n,k;

    // DCT99
    for(n=0;n<9;n++)
        for(k=0;k<9;k++)
            Cos99[n][k]=cos(M_PI/18*(2*k+1)*n);

    // for DCT99W
    Cos99W[1]=Cos99[1][7];
    Cos99W[2]=Cos99[2][0];
    Cos99W[3]=Cos99[2][2];
    Cos99W[4]=Cos99[2][3];
    Cos99W[5]=Cos99[2][1];
    Cos99W[6]=Cos99[1][5];
    Cos99W[7]=Cos99[1][8];
    Cos99W[8]=Cos99[1][6];

    // DCT1818
    for(n=0;n<18;n++)
        for(k=0;k<18;k++)
            Cos1818[n][k]=cos(M_PI/36*(2*k+1)*n);
    // DCT1818F
}

for(k=0;k<18;k++)
    Cos1818F[k]=2*Cos1818[1][k];

// DCT3618
for(n=0;n<18;n++)
    for(k=0;k<36;k++)
        Cos3618[n][k]=cos(M_PI/72*(2*k+19)*(2*n+1));

// DCT3618F
for(k=0;k<18;k++)
    Cos3618F[k]=2*cos(M_PI/72*(2*k+1));

// IMDCT1836F
for(k=0;k<18;k++)
    Cos1836F[k]=cos(M_PI/72*19*(2*k+1));
}

=====

=====
// 9-9 DCT, standard
=====
void DCT99(double *a, double *A)
{
    int n,k;
    double s;

    for(n=0;n<9;n++)
    {
        s=0;
        for(k=0;k<9;k++)
        {
            s+=a[k]*Cos99[n][k];
        }
        A[n]=s;
    }
}

=====

// 9-9 DCT, Winograd
=====
void DCT99W(double *a, double *A)
{
    double d[64],m[64];

    d[1]=a[3]+a[5];
    d[2]=a[3]-a[5];
    d[3]=a[6]+a[2];
    d[4]=a[6]-a[2];
    d[5]=a[1]+a[7];
    d[6]=a[1]-a[7];
    d[7]=a[8]+a[0];
    d[8]=a[8]-a[0];
    d[ 9]=a[ 4]+d[5];
    d[10]=d[ 1]+d[3];
    d[11]=d[10]+d[7];
    d[12]=d[ 3]-d[7];
    d[13]=d[ 1]-d[7];
}

```

```

d[14]=d[ 1]-d[3];
d[15]=d[ 2]-d[4];
d[16]=d[15]+d[8];
d[17]=d[ 4]+d[8];
d[18]=d[ 2]-d[8];
d[19]=d[ 2]+d[4];

m[ 1]=Cos99W[1]*d[ 6];
m[ 2]=Cos99W[5]*d[ 5];
m[ 3]=Cos99W[5]*d[11];
m[ 4]=Cos99W[2]*d[12];
m[ 5]=Cos99W[3]*d[13];
m[ 6]=Cos99W[4]*d[14];
m[ 7]=Cos99W[1]*d[16];
m[ 8]=Cos99W[6]*d[17];
m[ 9]=Cos99W[7]*d[18];
m[10]=Cos99W[8]*d[19];

d[20]=a[ 4]-m[2];
d[21]=d[20]+m[4];
d[22]=d[20]-m[4];
d[23]=d[20]+m[5];
d[24]=m[ 1]+m[8];
d[25]=m[ 1]-m[8];
d[26]=m[ 1]+m[9];

A[0]=d[ 9]+d[11];
A[1]=m[10]-d[26];
A[2]=m[ 6]-d[21];
A[3]=m[ 7];
A[4]=d[22]-m[ 5];
A[5]=d[25]-m[ 9];
A[6]=m[ 3]-d[ 9];
A[7]=d[24]+m[10];
A[8]=d[23]+m[ 6];
}

//=====

// 18-18 DCT, standard
//=====
void DCT1818(double *a, double *A)
{
int n,k;
double s;

for(n=0;n<18;n++)
{
s=0;
for(k=0;k<18;k++)
{
s+=a[k]*Cos1818[n][k];
}
A[n]=s;
}
}

//=====

// 18-18 DCT, fast
//=====
void DCT1818F(double *x, double *X)
{
int n,k;
double a[9],b[9],A[9],B[9],Bp[9];

for(k=0;k<9;k++)
{
a[k]=x[k]+x[18-1-k];
b[k]=(x[k]-x[18-1-k])*Cos1818F[k];
}
DCT99W(a,A);
DCT99W(b,Bp);

B[0]=Bp[0]/2;
for(n=1;n<9;n++) B[n]=Bp[n]-B[n-1];

for(n=0;n<9;n++)
{
X[2*n]=A[n];
X[2*n+1]=B[n];
}
}
//=====

//=====

// 36-18 DCT, standard
//=====
void MDCT3618(double *a, double *A)
{
int n,k;
double s;

for(n=0;n<18;n++)
{
s=0;
for(k=0;k<36;k++)
{
s+=a[k]*Cos3618[n][k];
}
A[n]=s;
}
}

//=====

// 36-18 DCT, fast
//=====
void MDCT3618F(double *y, double *Y)
{
int n,k;
double x[18],Yp[18];

for(k=0;k<9;k++)
x[k]=(-y[26-k]-y[27+k])*Cos3618F[k];
for(k=9;k<18;k++)
x[k]=( y[k-9 ]-y[26-k])*Cos3618F[k];

DCT1818F(x,Yp);
Y[0]=Yp[0]/2;
for(n=1;n<18;n++) Y[n]=Yp[n]-Y[n-1];
}
}

```

```

//=====
// 18-36 IMDCT, standard
//=====
void IMDCT1836(double *Y, double *y)
{
int n,k;
double s;

for(n=0;n<36;n++)
{
    s=0;
    for(k=0;k<18;k++)
    {
        s+=Y[k]*Cos3618[k][n];//share
with the same table as 3618
    }
    y[n]=s;
}
//=====

//=====
// 36-18 DCT, fast
//=====
void IMDCT1836F(double *Y, double *y)
{
int n,k;
double Yp[18],yppp[18],yp[36],s;

for(k=0;k<18;k++)
    Yp[k]=Y[k]*Cos3618F[k];//the      same
table

DCT1818F(Yp,yppp);

for(n= 0;n< 9;n++) yp[n]= yppp[n+9];
for(n= 9;n<10;n++) yp[n]= 0;
for(n=10;n<27;n++) yp[n]=-yppp[27-n];
for(n=27;n<36;n++) yp[n]=-yppp[n-27];

s=0;
//for(k=0;k<18;k++) s+=Yp[k];
for(k=0;k<18;k++) s+=Y[k]*Cos1836F[k];
y[0]=s;
for(n=1;n<36;n++) y[n]=yp[n]-y[n-1];
}
//=====
//*****FILE name: dct_lib.h*****
//*****FILE name: dct_lib.h*****
//*****FILE name: dct_lib.h*****
```